An Analysis of Bayesian Classifiers

Pat Langley **Wayne Iba*** **Kevin Thompson†**
{LANGLEY,IBA,KTHOMPSO}@PTOLEMY.ARC.NASA.GOV
AI Research Branch (M/S 269-2)
NASA Ames Research Center
Moffett Field, CA 94035 USA

Abstract

In this paper we present an average-case analysis of the Bayesian classifier, a simple induction algorithm that fares remarkably well on many learning tasks. Our analysis assumes a monotone conjunctive target concept, and independent, noise-free Boolean attributes. We calculate the probability that the algorithm will induce an arbitrary pair of concept descriptions and then use this to compute the probability of correct classification over the instance space. The analysis takes into account the number of training instances, the number of attributes, the distribution of these attributes, and the level of class noise. We also explore the behavioral implications of the analysis by presenting predicted learning curves for artificial domains, and give experimental results on these domains as a check on our reasoning.

Probabilistic Approaches to Induction

One goal of research in machine learning is to discover principles that relate algorithms and domain characteristics to behavior. To this end, many researchers have carried out systematic experimentation with natural and artificial domains in search of empirical regularities (e.g., Kibler & Langley, 1988). Others have focused on theoretical analyses, often within the paradigm of probably approximately correct learning (e.g., Haussler, 1990). However, most experimental studies are based only on informal analyses of the learning task, whereas most formal analyses address the worst case, and thus bear little relation to empirical results.

A third approach, proposed by Cohen and Howe (1988), involves the formulation of average-case models for specific algorithms and testing them through experimentation. Pazzani and Sarrett's (1990) study of conjunctive learning provides an excellent example of this technique, as does Hirschberg and Pazzani's (1991) work on inducing k-CNF concepts. By assuming information about the target concept, the number of attributes, and the class and attribute frequencies, they obtain predictions about the behavior of induction algorithms and used experiments to check their analyses.[1] However, their research does not focus on algorithms typically used by the experimental and practical sides of machine learning, and it is important that average-case analyses be extended to such methods.

Recently, there has been growing interest in probabilistic approaches to inductive learning. For example, Fisher (1987) has described COBWEB, an incremental algorithm for conceptual clustering that draws heavily on Bayesian ideas, and the literature reports a number of systems that build on this work (e.g., Allen & Langley, 1990; Iba & Gennari, 1991; Thompson & Langley, 1991). Cheeseman et al. (1988) have outlined AUTOCLASS, a nonincremental system that uses Bayesian methods to cluster instances into groups, and other researchers have focused on the induction of Bayesian inference networks (e.g., Cooper & Kerskovits, 1991).

These recent Bayesian learning algorithms are complex and not easily amenable to analysis, but they share a common ancestor that is simpler and more tractable. This supervised algorithm, which we refer to simply as a *Bayesian classifier*, comes originally from work in pattern recognition (Duda & Hart, 1973). The method stores a probabilistic summary for each class; this summary contains the conditional probability of each attribute value given the class, as well as the probability (or base rate) of the class. This data structure approximates the representational power of a perceptron; it describes a single decision boundary through the instance space. When the algorithm encounters a new instance, it updates the probabilities stored with the specified class. Neither the order of training instances nor the occurrence of classification errors have any effect on this process. When given a test instance, the classifier uses an evaluation function (which we describe in detail later) to rank the alter-

*Also affiliated with RECOM Technologies
†Also affiliated with Sterling Software.

[1] A related approach involves deriving the optimal learning algorithm under certain assumptions, and then implementing an approximation of that algorithm (e.g., Opper & Haussler, 1991).

DOMAIN	BAYES	IND/C4	FREQ.
SOYBEAN	100.0 ± 0.0	93.8 ± 3.4	36.2
CHESS	87.5 ± 0.4	99.3 ± 0.1	52.2
LYMPHO	81.1 ± 1.8	74.8 ± 2.2	56.7
SPLICE	94.6 ± 0.4	89.2 ± 0.5	53.2
PROMOTERS	87.4 ± 2.2	74.5 ± 1.9	50.0

Table 1: Percentage accuracies for two induction algorithms on five classification domains, along with the accuracy of predicting the most frequent class.

native classes based on their probabilistic summaries, and assigns the instance to the highest scoring class.

Both the evaluation function and the summary descriptions used in Bayesian classifiers assume that attributes are statistically independent. Since this seems unrealistic for many natural domains, researchers have often concluded that the algorithm will behave poorly in comparison to other induction methods. However, no studies have examined the extent to which violation of this assumption leads to performance degradation, and the probabilistic approach should be quite robust with respect to both noise and irrelevant attributes. Moreover, earlier studies (e.g., Clark & Niblett, 1987) present evidence of the practicality of the algorithm.

Table 1 presents additional experimental evidence for the utility of Bayesian classifiers. In this study we compare the method to IND's emulation of the C4 algorithm (Buntine & Caruana, 1991) and an algorithm that simply predicts the modal class. The five domains, from the UCI database collection (Murphy & Aha, 1992), include the "small" soybean dataset, chess end games involving a king-rook-king-pawn confrontation, cases of lymphography diseases, and two biological datasets. For each domain, we randomly split the data set into 80% training instances and 20% test instances, repeating this process to obtain 50 separate pairs of training and test sets. The table shows the mean accuracy and 95% confidence intervals on the test sets for each domain.

In four of the domains, the Bayesian classifier is at least as accurate as the C4 reimplementation. We will not argue that the Bayesian classifier is superior to this more sophisticated method, but the results do show that it behaves well across a variety of domains. Thus, the Bayesian classifier is a promising induction algorithm that deserves closer inspection, and a careful analysis should give us insights into its behavior.

We simplify matters by limiting our analysis to the induction of conjunctive concepts. Furthermore, we assume that there are only two classes, that each attribute is Boolean, and that attributes are independent of each other. We divide our study into three parts. We first determine the probability that the algorithm will learn a particular pair of concept descriptions. After this, we derive the accuracy of an arbitrary pair of descriptions over all instances. Taken together, these expressions give us the overall accuracy of the learned concepts. We find that a number of factors influence behavior of the algorithm, including the number of training instances, the number of relevant and irrelevant attributes, the amount of class and attribute noise, and the class and attribute frequencies. Finally, we examine the implications of the analysis by predicting behavior in specific domains, and check our reasoning with experiments in these domains.

Probability of Induced Concepts

Consider a concept C defined as the monotone conjunction of r relevant features $A_1, \ldots, A_r$ (i.e., in which none of the features are negated). Also assume there are i irrelevant features $A_{r+1}, \ldots, A_{r+i}$. Let $P(A_j)$ be the probability of feature A_j occurring in an instance. The concept descriptions learned by a Bayesian classifier are fully determined by the n training instances it has observed. Thus, to compute the probability of each such concept description, we must consider different possible combinations of n training instances.

First let us consider the probability that the algorithm has observed exactly k out of n positive instances. If we let $P(C)$ be the probability of observing a positive instance and we let x be the observed fraction of positive instances, then we have

$$P(x = \frac{k}{n}) = \binom{n}{k} P(C)^k [1 - P(C)]^{n-k} \quad .$$

This expression also represents the probability that one has observed exactly $n - k$ negative instances. Since we assume that the concept is monotone conjunctive and that the attributes are independent, we have $P(C) = \prod_{j=1}^{r} P(A_j)$, which is simply the product of the probabilities for all relevant attributes.

A given number of positive instances k can produce many alternative descriptions of the positive class, depending on the instances that are observed. One can envision each such concept description as a cell in an $r + i$ dimensional matrix, with each dimension ranging from 0 to k, and with the count on dimension j representing the number of positive instances in which attribute A_j was present. One can envision a similar matrix for the negative instances, again having dimensionality $r+i$, but with each dimension ranging from 0 to $n - k$, and with the count on each dimension j representing the number of negative instances in which A_j occurred. Figure 1 shows a positive cell matrix with $r + i = 3$, $k = 2$. The designated cell holds the probability that the algorithm has seen two instances with A_1 present, 1 instance with A_2 present, and 0 instances with A_3 present.

In both matrices, one can index each cell or concept description by a vector of length $r + i$. Let $P(cell_{\vec{a}})_k$ be the probability that the algorithm has produced the

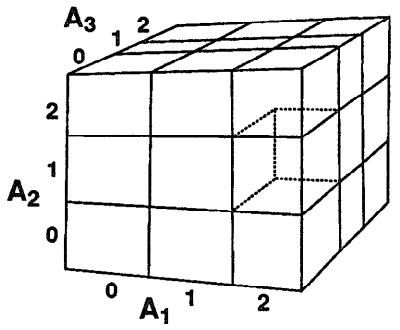

Figure 1: A positive cell matrix for three attributes and $k = 2$. Values along axes represent numbers of positive instances for which A_j was present.

cell indexed by vector $\vec{u}$ in the positive matrix given k positive instances; let $P(cell_{\vec{v}})_{n-k}$ be the analogous probability for a cell in the negative matrix. Then a weighted product of these terms gives the probability that the learning algorithm will generate any particular pair of concept descriptions, which is

$$P(k, \vec{u}, \vec{v})_n = P(x = \frac{k}{n}) P(cell_{\vec{u}})_k P(cell_{\vec{v}})_{n-k} \quad .$$

In other words, one multiplies the probability of seeing k out of n positive instances and the probabilities of encountering cell $\vec{u}$ in the positive matrix and cell $\vec{v}$ in the negative matrix.

However, we must still determine the probability of a given cell from the matrix. For those in the positive matrix, this is straightforward, since the attributes remain independent when the instance is a member of a conjunctive concept. Thus, we have

$$P(cell_{\vec{u}})_k = \prod_{j=1}^{r+i} P(y_j = \frac{\vec{u}_j}{k})$$

as the probability for $cell_{\vec{u}}$ in the positive matrix, where y_j represents the observed fraction of the k instances in which attribute A_j was present. Furthermore, the probability that one will observe A_j in exactly $\vec{u}_j$ out of k such instances is

$$P(y = \frac{\vec{u}_j}{k}) = \binom{k}{\vec{u}_j} P(A_j|C)^{\vec{u}_j} [1 - P(A_j|C)]^{k-\vec{u}_j} \quad .$$

In the absence of noise, we have $P(A_j|C) = 1$ for all relevant attributes and $P(A_j|C) = P(A_j)$ for all irrelevant attributes.

The calculation is more difficult for cells in the negative matrix. One cannot simply take the product of the probabilities for each index of the cell, since for a conjunctive concept, the attributes are not statistically independent. However, one can compute the probability that the $n - k$ observed negative instances will be composed of a particular combination of instances.

If we let $P(I_j|\bar{C})$ be the probability of I_j given a negative instance, we can use the multinomial distribution to compute the probability that exactly d_1 of the $n - k$ instances will be instance I_1, d_2 will be instance I_2, ..., and d_w will be instance I_w. Thus the expression

$$\frac{(n-k)!}{d_1! d_2! \ldots d_w!} P(I_1|\bar{C})^{d_1} P(I_2|\bar{C})^{d_2} \ldots P(I_w|\bar{C})^{d_w}$$

gives us the probability of a particular combination of negative instances, and from that combination we can compute the concept description (i.e., cell indices) that result. Of course, two or more combinations of instances may produce the same concept description, but one simply sums the probabilities for all such combinations to get the total probability for the cell. All that we need to make this operational is $P(I_j|\bar{C})$, the probability of I_j given a negative instance. In the absence of noise, this is simply $P(I_j)/P(\bar{C})$, since $P(\bar{C}|I_j) = 1$.

We can extend the framework to handle class noise by modifying the definitions of three basic terms – $P(C)$, $P(A_j|C)$, and $P(I_j|\bar{C})$. One common definition of class noise involves the corruption of class names (i.e., replacing the actual class with its opposite) with a certain probability z between 0 and 1. The probability of the class after one has corrupted values is

$$P'(C) = (1-z)P(C) + z(1-P(C)) = P(C)[1-2z] + z \quad ,$$

as we have noted elsewhere (Iba & Langley, 1992).

For an irrelevant attribute A_j, the probability $P(A_j|C)$ is unaffected by class noise and remains equal to $P(A_j)$, since the attribute is still independent of the class. However, the situation for relevant attributes is more complicated. By definition, we can reexpress the corrupted conditional probability of a relevant attribute A_j given the (possibly corrupted) class C as

$$P'(A_j|C) = \frac{P'(A_j \wedge C)}{P'(C)} \quad ,$$

where $P'(C)$ is the noisy class probability given above. Also, we can rewrite the numerator to specify the situations in which corruption of the class name does and does not occur, giving

$$P'(A_j|C) = \frac{(1-z)P(C)P(A_j|C) + zP(\bar{C})P(A_j|\bar{C})}{P'(C)} \quad .$$

Since we know that $P(A_j|C) = 1$ for a relevant attribute, and since $P(A_j|\bar{C}) = [P(A_j) - P(C)]/P(\bar{C})$ for conjunctive concepts, we have

$$P'(A_j|C) = \frac{(1-z)P(C) + z[P(A_j) - P(C)]}{P(C)[1-2z] + z} \quad ,$$

which involves only terms that existed before corruption of the class name.

We can use similar reasoning to compute the post-noise probability of any particular instance given that it is negative. As before, we can rewrite $P'(I_j|\bar{C})$ as

$$\frac{P'(I_j \wedge \bar{C})}{P'(\bar{C})} = \frac{(1-z)P(\bar{C})P(I_j|\bar{C}) + zP(C)P(I_j|C)}{1 - (P(C)[1-2z] + z)} \quad ,$$

but in this case the special conditions are somewhat different. For a negative instance, we have $P(I_j|C) = 0$, so that the second term in the numerator becomes zero. In contrast, for a positive instance, we have $P(I_j|\bar{C}) = 0$, so that the first term disappears. Taken together, these conditions let us generate probabilities for cells in the negative matrix after one has added noise to the class name.

After replacing $P(C)$ with $P'(C)$, $P(A_j|C)$ with $P'(A_j|C)$, and $P(I_j|\bar{C})$ with $P'(I_j|\bar{C})$, the expressions earlier in this section let us compute the probability that a Bayesian classifier will induce any particular pair of concept descriptions (cells in the two matrices). The information necessary for this calculation is the number of training instances, the number of relevant and irrelevant attributes, their distributions, and the level of class noise. This analysis holds only for monotone conjunctive concepts and in domains with independent attributes, but many of the ideas should carry over to less restricted classes of domains.

Accuracy of Induced Concepts

To calculate overall accuracy after n training instances, we must sum the expected accuracy for each possible instance weighted by that instance's probability of occurrence. More formally, the expected accuracy is

$$K_n = \sum_{j}^{I} P(I_j) K(I_j)_n \quad .$$

To compute the expected accuracy $K(I_j)_n$ for instance I_j, we must determine, for each pair of cells in the positive and negative matrices, the instance's classification.

A test instance I_j is classified by computing its score for each class description and selecting the class with the highest score (choosing randomly in case of ties). We will define $accuracy(I_j)_{n,k,\vec{u},\vec{v}}$ for the pair of concept descriptions $\vec{u}$ and $\vec{v}$ to be 1 if this scheme correctly predicts I_j's class, 0 if it incorrectly predicts the class, and $\frac{1}{2}$ if a tie occurs.

Following our previous notation, let n be the number of observed instances, k be the number of observed positive instances, u_j be the number of positive instances in which attribute A_j occurs, and v_j be the number of negative instances in which A_j occurs. For a given instance I_j, one can compute the score for the positive class description as

$$score(C)_j = \frac{k}{n} \prod_{j=1}^{r+i} \begin{cases} \frac{u_j}{k} & \text{if } A_j \text{ is present in } I_j \\ \frac{k-u_j}{k} & \text{otherwise,} \end{cases}$$

and an analogous equation for the negative class, substituting $n-k$ for k and v for u. To avoid multiplying by 0 when an attribute has never (always) been observed in the training instances but is (is not) present in the test instance, we follow Clark and Niblett's (1987) suggestion of replacing 0 with a small value, such as $1/2n$.

To compute the expected accuracy for instance I_j, we sum, over all possible values of k and pairs of concept descriptions, the product of the probability of selecting the particular pair of concept descriptions after k positive instances and the pair's accuracy on I_j. Thus, we have

$$K(I_j)_n = \sum_{k=0}^{n} \sum_{\vec{u}}^{\mathbf{U}} \sum_{\vec{v}}^{\mathbf{V}} P(k, \vec{u}, \vec{v})_n \; accuracy(I_j)_{n,k,\vec{u},\vec{v}} \quad ,$$

where the second and third summations occur over the possible vectors that index into the positive matrix **U** and the negative matrix **V**. To complete our calculations, we need an expression for $P(I_j)$, which is the product of the probabilities of features present in I_j.

Implications for Learning Behavior

Although the equations in the previous sections give a formal description of the Bayesian classifier's behavior, their implications are not obvious. In this section, we examine the effects of various domain characteristics on the algorithm's classification accuracy. However, because the number of possible concept descriptions grows exponentially with the number of training instances and the number of attributes, our predictions have been limited to a small number of each.

In addition to theoretical predictions, we report learning curves that summarize runs on 100 randomly generated training sets. Each curve reports the average classification accuracy over these runs on a single test set of 200 randomly generated instances containing no noise. In each case, we bound the mean accuracy with 95% confidence intervals to show the degree to which our predicted learning curves fit the observed ones. These experimental results provide an important check on our reasoning, and they revealed a number of problems during development of the analysis.

Figure 2 (a) shows the effects of concept complexity on the rate of learning in the Bayesian classifier when no noise is present. In this case, we hold the number of irrelevant attributes i constant at one, and we hold their probability of occurrence $P(A)$ constant at $\frac{1}{2}$. We vary both the number of training instances and the number of relevant attributes r, which determine the complexity of the target concept. To normalize for effects of the base rate, we also hold $P(C)$, the probability of the concept, constant at $\frac{1}{2}$; this means that, for each of the r relevant attributes, $P(A)$ is $P(C)^{1/r}$, and thus is varied for the different conditions.[2]

As typical with learning curves, the initial accuracies begin low (at $\frac{1}{2}$) and gradually improve with increasing numbers of training instances. The effect of concept complexity also agrees with our intuitions; introducing

[2] An alternative approach would hold $P(A)$ constant for relevant attributes, causing $P(C)$ to become $P(A)^r$. This nudges the initial accuracies upward but otherwise has little effect on the learning curves.

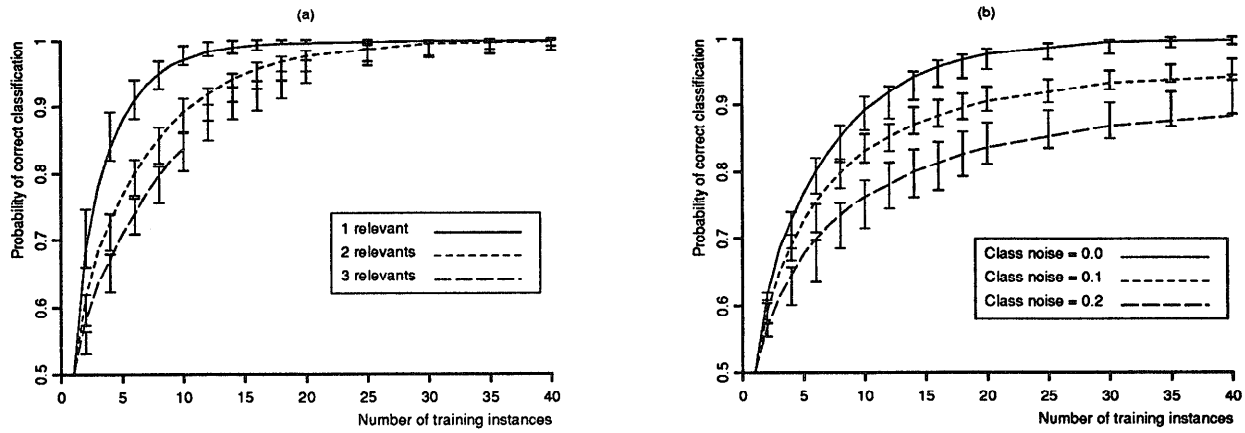

Figure 2: Predictive accuracy of a Bayesian classifier in a conjunctive concept, assuming the presence of one irrelevant attribute, as a function of training instances and (a) number of relevant attributes and (b) amount of class noise. The lines represent theoretical learning curves, whereas the error bars indicate experimental results.

additional features into the target concept slows the learning rate, but does not affect asymptotic accuracy, which is always 1.0 for conjunctive concepts on noise-free test cases. The rate of learning appears to degrade gracefully with increasing complexity. The predicted and observed learning curves are in close agreement, which lends confidence to our average-case analysis. Theory and experiment show similar effects when we vary the number of irrelevant attributes; learning rate slows as we introduce misleading features, but the algorithm gradually converges on perfect accuracy.

Figure 2 (b) presents similar results on the interaction between class noise and the number of training instances. Here we hold the number of relevant attributes constant at two and the number of irrelevants constant at one, and we examine three separate levels of class noise. Following the analysis, we assume the test instances are free of noise, which normalizes accuracies and eases comparison. As one might expect, increasing the noise level z decreases the rate of learning. However, the probabilistic nature of the Bayesian classifier leads to graceful degradation, and asymptotic accuracy should be unaffected. We find a close fit between the theoretical behavior and the experimental learning curves. Although our analysis does not incorporate attribute noise, experiments with this factor produce similar results. In this case, equivalent levels lead to somewhat slower learning rates, as one would expect given that attribute noise can corrupt multiple values, whereas class noise affects only one.

Finally, we can compare the behavior of the Bayesian classifier to that of WHOLIST (Pazzani & Sarrett, 1990). One issue of interest is the number of training instances required to achieve some criterion level of accuracy. A quantitative comparison of this nature is beyond the scope of this paper, but the respective analyses and experiments show that the WHOLIST algorithm is only affected by the number of irrelevant attributes, whereas the Bayesian classifier is sensitive to both the number of relevant and irrelevant attributes. However, the Bayesian classifier is robust with respect to noise, whereas the WHOLIST algorithm is not.

Discussion

In this paper we have presented an analysis of a Bayesian classifier. Our treatment requires that the concept be monotone conjunctive, that instances be free of attribute noise, and that attributes be Boolean and independent. Given information about the number of relevant and irrelevant attributes, their frequencies, and the level of class noise, our equations compute the expected classification accuracy after a given number of training instances.

To explore the implications of the analysis, we have plotted the predicted behavior of the algorithm as a function of the number of training instances, the number of relevant attributes, and the amount of noise, finding graceful degradation as the latter two increased. As a check on our analysis, we run the algorithm on artificial domains with the same characteristics. We obtain close fits to the predicted behavior, but only after correcting several errors in our reasoning that the empirical studies revealed.

In additional experiments, we compare the behavior of the Bayesian classifier to that of a reimplementation of C4, a more widely used algorithm that induces decision trees. In general, the probabilistic method performs comparably to C4, despite the latter's greater sophistication. These results suggest that such simple methods deserve increased attention in future studies, whether theoretical or experimental.

In future work, we plan to extend this analysis in several ways. In particular, our current equations handle only class noise, but as Angluin and Laird (1988) have shown, attribute noise can be even more problematic for learning algorithms. We have developed

tentative equations for the case of attribute noise, but the expressions are more complex than for class noise, in that the possible corruption of any combination of attributes can make any instance appear like another. We also need to relax the constraint that target concepts must be monotone conjunctive.

Another direction in which we can extend the present work involves running additional experiments. Even within the assumptions of the current analysis, we could empirically study the extent to which violated assumptions alter the observed behavior of the algorithm. In addition, we could analyze the attribute frequencies in several of the domains commonly used in experiments to determine the analytic model's ability to predict behavior on these domains given their frequencies as input. This approach would extend the usefulness of our average-case model beyond the artificial domains on which we have tested it to date.

Overall, we are encouraged by the results that we have obtained. We have demonstrated that a simple Bayesian classifier compares favorably with a more sophisticated induction algorithm and, more important, we have characterized its average-case behavior for a restricted class of domains. Our analysis confirms intuitions about the robustness of the Bayesian algorithm in the face of noise and concept complexity, and it provides fertile ground for further research on this understudied approach to induction.

Acknowledgements

Thanks to Stephanie Sage, Kimball Collins, and Andy Philips for discussions that helped clarify our ideas.

References

Allen, J.A., & Langley, P. (1990). Integrating memory and search in planning. *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 301–312). San Diego: Morgan Kaufmann.

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning*, *2*, 343–370.

Buntine, W., & Caruana, R. (1991). *Introduction to IND and recursive partitioning* (Technical Report FIA-91-28). Moffett Field, CA: NASA Ames Research Center, Artificial Intelligence Research Branch.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–284.

Cohen, P. R., & Howe, A. E. (1988). How evaluation guides AI research. *AI Magazine*, *9*, 35–43.

Cooper, G. F., & Herskovits, E. (1991). A Bayesian method for constructing Bayesian belief networks from databases. *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence* (pp. 86–94). Los Angeles: Morgan Kaufmann.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Haussler, D. (1990). Probably approximately correct learning. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1101–1108). Boston: AAAI Press.

Iba, W., & Gennari, J. H. (1991). Learning to recognize movements. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo: Morgan Kaufmann.

Iba, W., & Langley, P. (1992). Induction of one-level decision trees. *Proceedings of the Ninth International Conference on Machine Learning*. Aberdeen: Morgan Kaufmann.

Hirschberg, D. S., & Pazzani, M. J. (1991). *Average-case analysis of a k-CNF learning algorithm* (Technical Report 91-50). Irvine: University of California, Department of Information & Computer Science.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow: Pittman.

Murphy, P. M., & Aha, D. W. (1992). *UCI Repository of machine learning databases* [Machine-readable data repository]. Irvine: University of California, Department of Information & Computer Science.

Opper, M., & Haussler. D. (1991). Calculation of the learning curve of Bayes optimal classification algorithm for learning a perceptron with noise. *Proceedings of the Fourth Annual Workshop on Computational Learning Theory* (pp. 75–87). Santa Cruz: Morgan Kaufmann.

Pazzani, M. J., & Sarrett, W. (1990). Average-case analysis of conjunctive learning algorithms. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 339–347). Austin, TX: Morgan Kaufmann.

Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo: Morgan Kaufmann.